MANUAL DE USUARIO
MANUAL DE CACHES DE SEGUNDO NIVEL EN HIBERNATE

Versión 1.0

Área de Aplicaciones Especiales y Arquitectura de Software

Agencia de Informática y Comunicaciones de la Comunidad de Madrid

c

Hoja de Control

Título	Manual de cachés de segundo nivel en Hibernate		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	Unidad de Arquitectura de Software de Aplicaciones		
Versión	1.0	**Fecha Versión**	16/06/2014

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Unidad de Arquitectura de Software	16/06/2014

Índice

1 INTRODUCCIÓN 4

1.1 AUDIENCIA OBJETIVO 4
1.2 CONOCIMIENTOS PREVIOS 4

2 DESCRIPCIÓN 5

3 INSTALACIÓN Y CONFIGURACIÓN 6

3.1 INSTALACIÓN 6
3.2 CONFIGURACIÓN 6
3.2.1 Activación del listener de Debug 6

4 USO DE LAS CACHÉS DE SEGUNDO NIVEL 7

4.1 ANOTACIÓN @CACHE 7
4.2 ANOTACIÓN @ATLASCACHE 8
4.3 CACHEO DE ENTIDADES OBTENIDAS POR PRIMARY KEY 9
4.4 CACHEO DE CONSULTAS A BASE DE DATOS MEDIANTE ATLASQUERY 9
4.5 CACHEO DE CONSULTAS A BASE DE DATOS MEDIANTE CRITERIA DE HIBERNATE 9

5 RECOMENDACIONES Y BUENAS PRACTICAS 10

6 ENLACES RELACIONADOS 10

1 INTRODUCCIÓN

Este manual explica el uso de la caché de segundo nivel de Hibernate en las aplicaciones ATLAS de los Sistemas de Información de la Comunidad de Madrid.

La implementación actual de la caché de segundo nivel de Atlas está basada en Ehcache (http://ehcache.org).

Atención
Las cachés de segundo nivel están prohibidas en los proyectos estándar de ICM por la norma **ADCache2nivel** Si se necesita el uso de las cachés en un proyecto primero se tendrá que hablar con Arquitectura de Software para estudiar cómo se van a implementar en el proyecto concreto

1.1 Audiencia objetivo

Este documento está dirigido a desarrolladores de proyectos java para ICM en los que se desee incorporar una cachés de segundo nivel a ciertas entidades de su aplicación.

1.2 Conocimientos Previos

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- Spring Framework.
- Hibernate
- Bases de datos Oracle

2 DESCRIPCIÓN

El Servicio de Cachés de Atlas está diseñado para evitar al desarrollador la labor de configurar las cachés de segundo nivel de Hibernate en una aplicación ATLAS. De esta forma y bajo una configuración sencilla el programador tendrá que "**marcar**" sus entidades de Hibernate como entidades cacheables y el Servicio de Caché se encargará de configurar y usar las cachés necesarias para las entidades y queries marcadas.

La caché de Atlas está pensada y optimizada para cachear aquellas **entidades que no sufran muchos cambios pero que sin embargo sí se lean muy frecuentemente (baja tasa de escritura y alta tasa de lectura)**. Es por ello que **no se recomienda el cacheo de objetos de negocio que estén sometidos a cambios frecuentes**. Algunos ejemplos de entidades cacheables son los **catálogos de las aplicaciones**, como los Municipios de la Comunidad de Madrid, los Tipos de Vía o el Sexo (V/M). Típicamente entidades cacheables se suelen mostrar sólo en componentes visuales como la Lista de Valores, o el Combo de Valores (un desplegable).

Atención
Las cachés se configuran para guardar los elementos sólo en memoria, sin ningún tipo de persistencia en disco. Es decir, cuando el servidor de aplicaciones se para o reinicia se borran todos los elementos de la caché. También se configuran sin replicación entre servidores del clúster. Es decir si una aplicación está montada en un clúster, cada servidor del clúster tendrá su propia caché en memoria independiente. Es muy importante tener esto en cuenta, porque **es muy probable que un cambio en base de datos en una entidad cacheada no se vea reflejado inmediatamente en una pantalla donde se muestre dicha entidad** (hasta que no se refresque la entidad de la caché no se verán los cambios, por esto hay que configurar muy bien los tiempos máximos de cacheo). Además, a algunos usuarios les puede aparecer la entidad actualizada y a otros no (dependiendo del servidor del clúster al que acceda cada uno de ellos).

Para utilizar la caché es necesario que la aplicación indique qué entidades y queries van a ser cacheadas. A la hora de indicar qué entidades son susceptibles de ser cacheadas, el programador deberá hacerlo usando la anotación **@Cache** propia de Hibernate junto con la anotación **@AtlasCache** propia de ATLAS, según se verá en los siguientes apartados.

3 INSTALACIÓN Y CONFIGURACIÓN

Los arquetipos no tienen activadas las cachés de segundo nivel de Hibernate por defecto. Por lo tanto aquellas aplicaciones que lo requieran tienen que realizar los siguientes pasos para poder utilizar el servicio correctamente:

3.1 INSTALACIÓN

Para usar las cachés de segundo nivel a partir de la versión 1.2.7 de Atlas no hay que instalar nada.

3.2 CONFIGURACIÓN

Las cachés de segundo nivel de Hibernate están configuradas por defecto en el Framework Atlas, pero su uso está prohibido hasta que no se tenga una autorización por parte de Arquitectura para usarlas en el proyecto

3.2.1 Activación del listener de Debug

Para comprobar el funcionamiento de las cachés y hacer un seguimiento más detallado de las mismas se puede activar un listener de debug que se mostrara un log de los datos que se insertan, borrar y actualizan en las cachés.

applicationContext-database.xml

```
<property name="hibernateProperties">
  <props>
   <prop key="hibernate.dialect">org.hibernate.dialect.Oracle9iDialect</prop>
   <prop key="hibernate.show_sql">false</prop>
   <prop key="hibernate.format_sql">false</prop>
   <prop key="hibernate.use_sql_comments">false</prop>
   <prop key="hibernate.cache.use_second_level_cache">true</prop>
   <prop key="hibernate.cache.use_query_cache">true</prop>
   <prop key="hibernate.cache.region.factory_class"
         >atlas.core.cache.AtlasCacheRegionFactory</prop>
   <prop key="hibernate.cache.region_prefix"></prop>
   <prop key="net.sf.ehcache.configurationResourceName">ehcache.xml</prop>
   <prop key="atlas.cache.debugListener">false</prop>
  </props>
</property>
```

Nota

Si se pone la variable **atlas.cache.debugListener** a true, se mostrará en el log, los datos que se insertan, borrar y actualizan en las cachés. Este parámetro puede estar bien para pruebas en local o desarrollo, pero es **muy importante dejar este valor a false cuando se pase a producción**.

4 USO DE LAS CACHÉS DE SEGUNDO NIVEL

En el desarrollo de una aplicación, y a la hora de crear las entidades de dominio, es cuando hay que identificar aquellas susceptibles de ser cacheadas e indicárselo al framework Atlas. Esto se hace anotando las entidades de dominio mediante las anotaciones **@Cache** (propia de Hibernate) y la anotación **@AtlasCache**.

A continuación se explican las 2 anotaciones:

4.1 ANOTACIÓN @Cache

Al anotar una entidad con ***@Cache***, indicamos a Hibernate que queremos que utilice las cachés de segundo nivel para esa entidad. De las propiedades de la anotación @Cache **solo se usará la propiedad usage** para indicar la estrategia de concurrencia de la caché. Esta propiedad puede tener los siguientes valores:

Usage	Explicación
Read-only	Es la estrategia de concurrencia menos estricta. Ideal para **datos que nunca cambian.**
Nonstrinct-read-write	**Se recomienda este tipo de cachés para entidades que sean catálogos o entidades de negocio que cambien poco.** No ofrece ninguna garantía de consistencia entre la caché y la base de datos. Para sincronizar los objetos de la caché con la base de datos se utilizan timeouts, de modo que cuando caduca el timeout se recargan los datos. Con esta estrategia se tiene un intervalo en el cual existe el riesgo de obtener objetos desfasados. Cuando Hibernate realiza una operación de flush() en una sesión, se invalidan los objetos de la caché de segundo nivel. Aún así, esta es una operación asíncrona y nunca se tienen garantías de que otro usuario no pueda leer datos erróneos. A pesar de todo esto, esta estrategia es ideal para almacenar datos que no sean demasiado críticos.
Read-write	**No se permite el uso de esta estrategia en las cachés de ATLAS** Mantiene un aislamiento hasta el nivel de commited, utilizando un sistema de marcas de tiempo (timestamps). El caso de uso recomendable es el mismo que para la estrategia transactional pero con la diferencia de que esta estrategia no se puede utilizar en entornos basados en clústeres.

Transactional	**No se permite el uso de esta estrategia en las cachés de ATLAS** Garantiza un nivel de aislamiento hasta repeatable read, si se necesita. Es el nivel más estricto. Se recomienda su uso cuando no se pueda permitir datos que queden desfasados. Esta estrategia sólo se puede utilizar en entornos basados en clústeres, es decir, con cachés distribuidas

4.2 ANOTACIÓN @AtlasCache

Propiedad	Explicación
maxEntities	Indica el número máximo de entidades a guardar en la memoria de la caché
segundosCache	Indica los segundos de vida de los elementos en la caché Valor por defecto: ***120***
maxQueries	Indica el número máximo de queries a guardar en la memoria de la caché de cada tipo de los especificados en las propiedades cacheSelectAll, cacheSelectWhere y cacheSelectLike
cacheSelectAll	Indica si se van a cachear aquellos queries o criterias que no tengan filtros. Valores posibles: true / false Valor por defecto: ***true***
cacheSelectWhere	Indica si se van a cachear aquellos queries o criterias que si tengan filtros, pero ninguno de ellos sea de tipo LIKE. Valores posibles: true / false Valor por defecto: ***true***
cacheSelectLike	Indica si se van a cachear aquellos queries o criterias que tengan filtros de tipo LIKE. Valores posibles: true / false Valor por defecto: ***false***

Ejemplo de uso de las anotaciones @Cache y @AtlasCache

Cliente.java

```
…

@Entity
@Table(name = "EJPL_CLIENTES")
@Cache(usage=CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
@AtlasCache(maxEntities=1000, segundosCache=120)
      public class Cliente implements java.io.Serializable {.....

…
```

4.3 Cacheo de entidades obtenidas por primary key

Todas las entidades que tengan la anotación @AtlasCache cachearán las entidades obtenidas mediante primary key automáticamente basándose en las propiedades **maxEntities** y **segundosCache** de la anotación.

4.4 Cacheo de consultas a Base de Datos mediante AtlasQuery

Las consultas a base de datos ejecutadas mediante la clase **AtlasQuery** comprueban si tienen que ser cacheadas en base a la anotación @AtlasCache.

Cuando se ejecuta una consulta a través de un @AtlasQuery se comprueba que tipo de filtro tiene esa consulta (Sin filtro, con filtros , o con filtros de tipo like) y en base a las propiedades de la anotación ***cacheSelectAll***, ***cacheSelectWhere*** y ***cacheSelectLike*** determina automáticamente si la query va a ser cacheada o no y activa la caché para la consulta.

4.5 Cacheo de consultas a Base de Datos mediante Criteria de Hibernate

Si se quieren cachear las consultas ejecutadas mediante Criterias de Hibernate se debe hacer a mano basándose en las regiones de cache creadas por la anotación @AtlasCache.
La anotación @AtlasCache crea 3 regiones para los queries de cada entidad a cachear con el nombre

- **query.all.<*entidad*>** para los queries que no tengan filtro. (Ejemplo: *query.all.Cliente*)
- **query.where.<*entidad*>** para los queries que tengan filtro y no sea LIKE. (Ejemplo: *query.where.Cliente*)
- **query.like.<*entidad*>** para los queries que tengan algún filtro LIKE. (Ejemplo: *query.like.Cliente*)

Cuando se vaya a ejecutar un Criteria y se quiera cachear se deberá hacer con los métodos **setCacheable()** y **setCacheRegion()** seleccionando el nombre de región adecuado.

ejemplo.java

```
…

    Criteria criteria = getHibernateTemplate().getSessionFactory().
        getCurrentSession().createCriteria(Cliente.class);
    criteria.add(Restrictions.eq("edad", 25));
    criteria.setCacheable(true);
    criteria.setCacheRegion("query.where.Cliente");
    return criteria.list();


…
```

5 RECOMENDACIONES Y BUENAS PRACTICAS

El uso de las cachés de segundo nivel en una aplicación debe ser estudiado conjuntamente por el área de Arquitectura y el responsable de la aplicación.

6 ENLACES RELACIONADOS

Producto	URL
Hibernate	http://www.hibernate.org
Ehcache	http://ehcache.org